Towards Distributed Multi-agent Optimization in a Stochastic Derivative-free Setting

Jeffrey Larson

Argonne National Laboratory

February 17, 2014

Background

2012: Ph.D. in Applied Mathematics from University of Colorado Denver

- Dissertation: Derivative-free Optimization of Noisy Functions

2012 - 2014: Postdoctoral Researcher, Department of Automatic Control, KTH Royal Institute of Technology

Present: Postdoctoral Researcher, Mathematics and Computer Science, Argonne National Laboratory

- Derivative-free Optimization
- Distributed Multi-agent Optimization
- Heavy-duty Vehicle Platooning
- Sports Scheduling
- Tiled QR Factorization

Background

2012: Ph.D. in Applied Mathematics from University of Colorado Denver

- Dissertation: Derivative-free Optimization of Noisy Functions

2012 - 2014: Postdoctoral Researcher, Department of Automatic Control, KTH Royal Institute of Technology

Present: Postdoctoral Researcher, Mathematics and Computer Science, Argonne National Laboratory

- **Derivative-free Optimization**
- **Distributed Multi-agent Optimization**
- Heavy-duty Vehicle Platooning
- Sports Scheduling
- Tiled QR Factorization

Contents

Outline

Stochastic Derivative-free Optimization

- Common Approaches
- Our Method
- Outline of Convergence Proof
- Numerical Results

Distributed Multi-agent Optimization

- Common Approaches
- Our Algorithm
- Outline of Convergence Proof
- Numerical Results

The Problem

We want to solve:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x)$$

when $\nabla f(x)$ is unavailable and we only have access to noise-corrupted function evaluations $\bar{f}(x)$.

The Problem

We want to solve:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x)$$

when $\nabla f(x)$ is unavailable and we only have access to noise-corrupted function evaluations $\bar{f}(x)$.

Such noise may be deterministic (e.g., from iterative methods) or stochastic (e.g., from a Monte-Carlo process).

The Problem

We want to solve:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x)$$

when $\nabla f(x)$ is unavailable and we only have access to noise-corrupted function evaluations $\bar{f}(x)$.

Such noise may be deterministic (e.g., from iterative methods) or stochastic (e.g., from a Monte-Carlo process).

Model-based methods are one of the most popular methods when ∇f is unavailable, and the only recourse when noise is deterministic.

The Problem

We analyze the convergence of our method in the stochastic case:

$$\bar{f}(x) = f(x) + \epsilon,$$

where ϵ is identically distributed with mean 0 and variance $\sigma^2 < \infty$.

The Problem

We analyze the convergence of our method in the stochastic case:

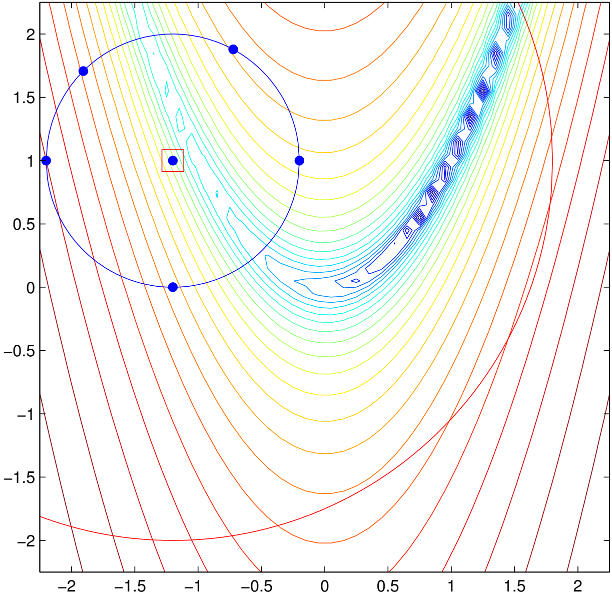
$$\bar{f}(x) = f(x) + \epsilon,$$

where ϵ is identically distributed with mean 0 and variance $\sigma^2 < \infty$.

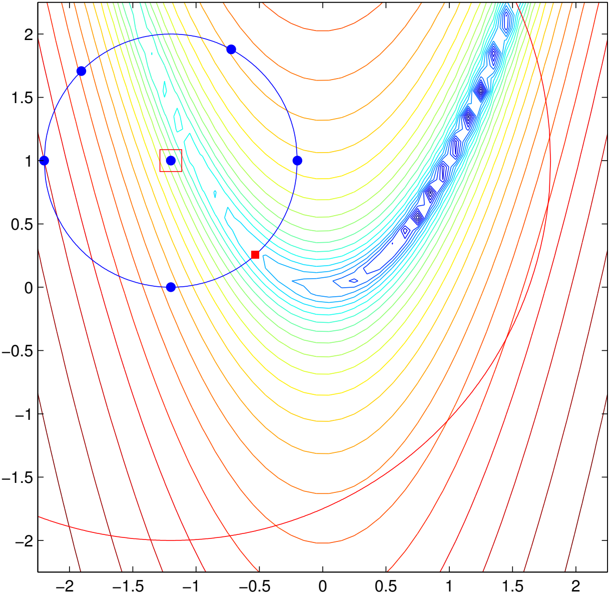
This is equivalent to solving:

$$\underset{x}{\text{minimize}}\ \mathbb{E}\left[\bar{f}(x)\right].$$

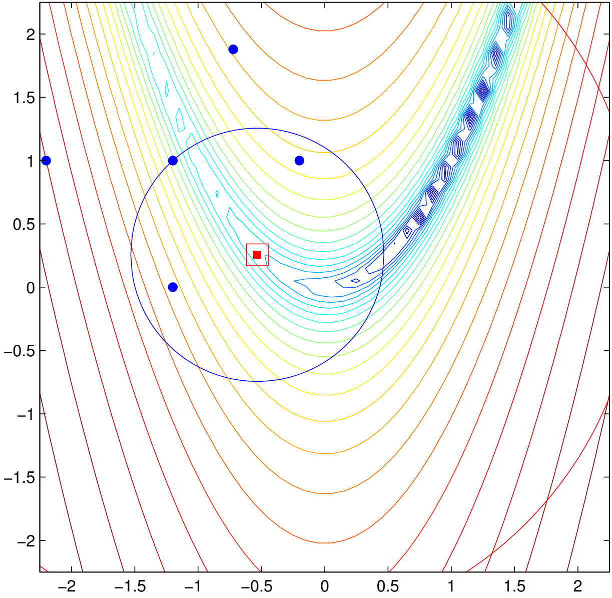
Prototype



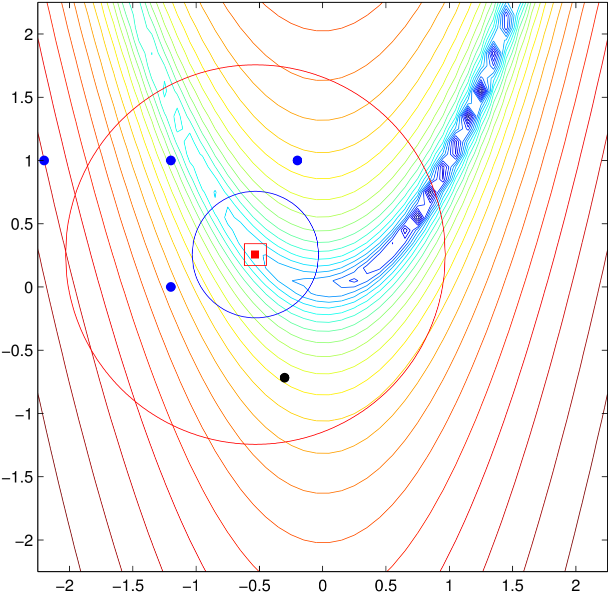
Prototype



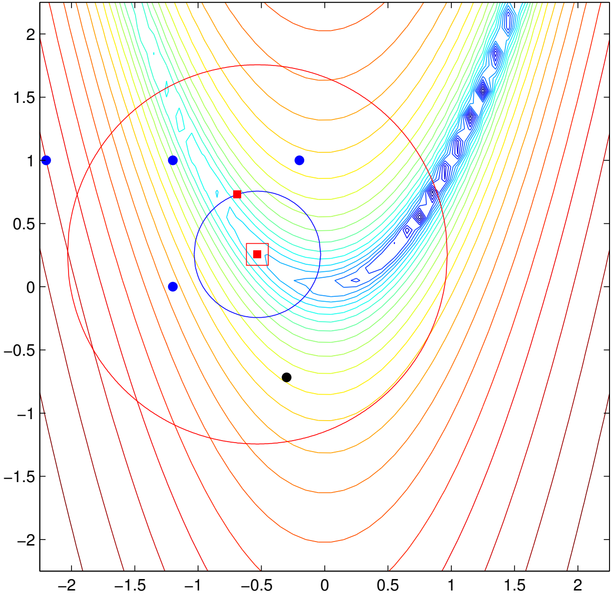
Prototype



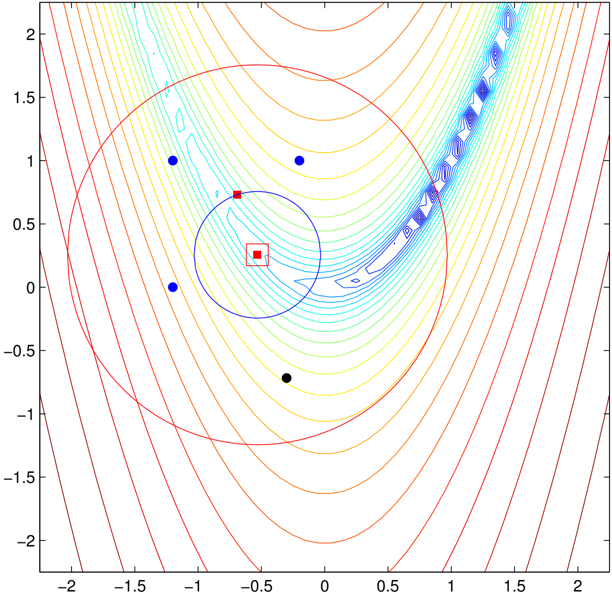
Prototype



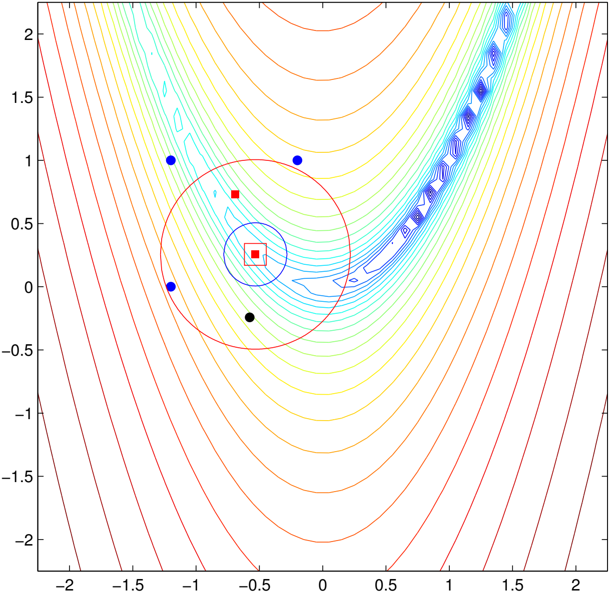
Prototype



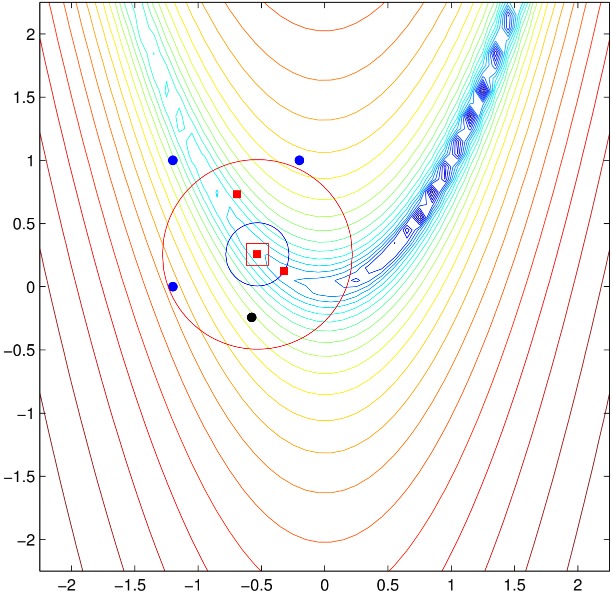
Prototype



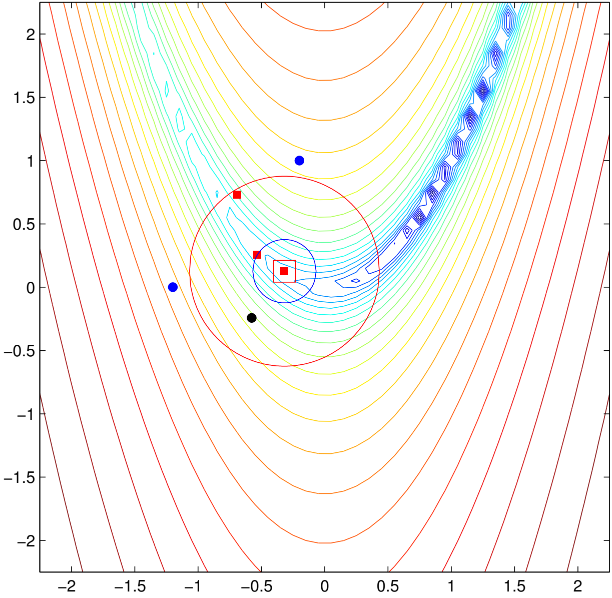
Prototype



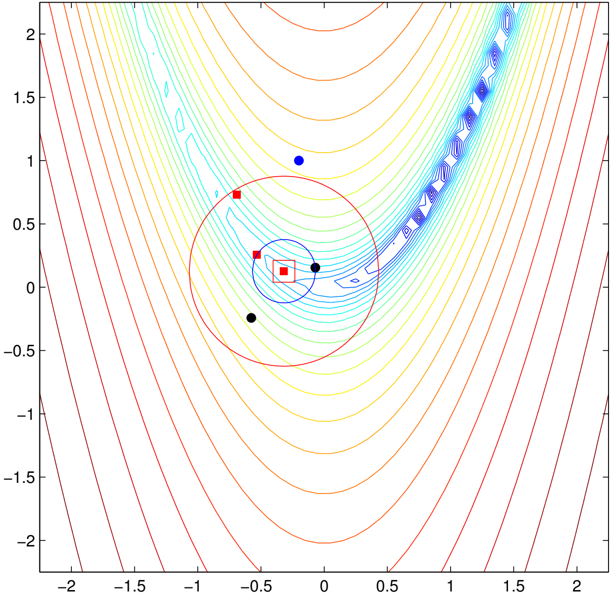
Prototype



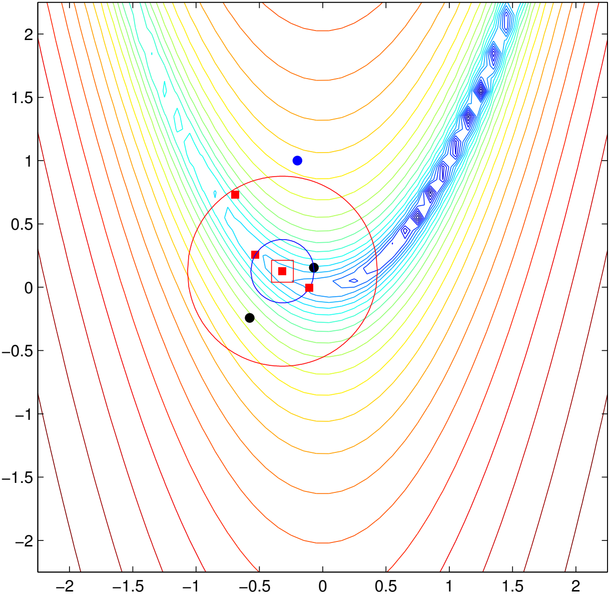
Prototype



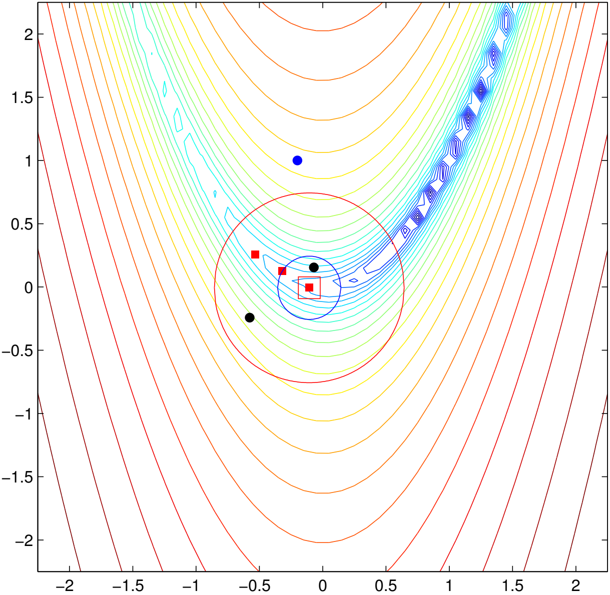
Prototype



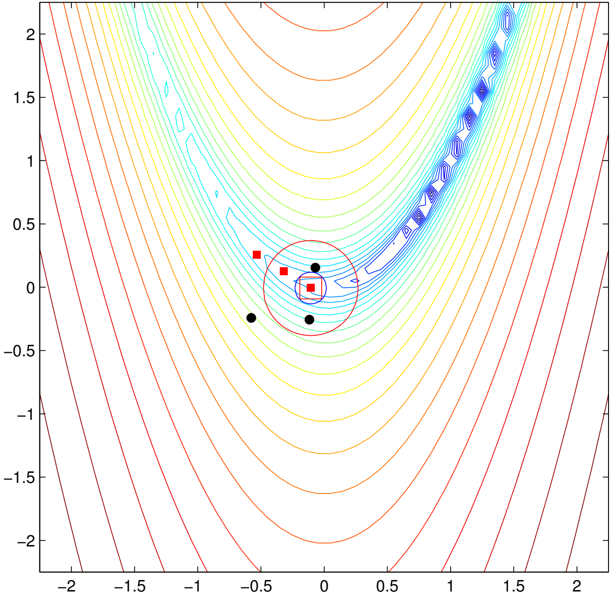
Prototype



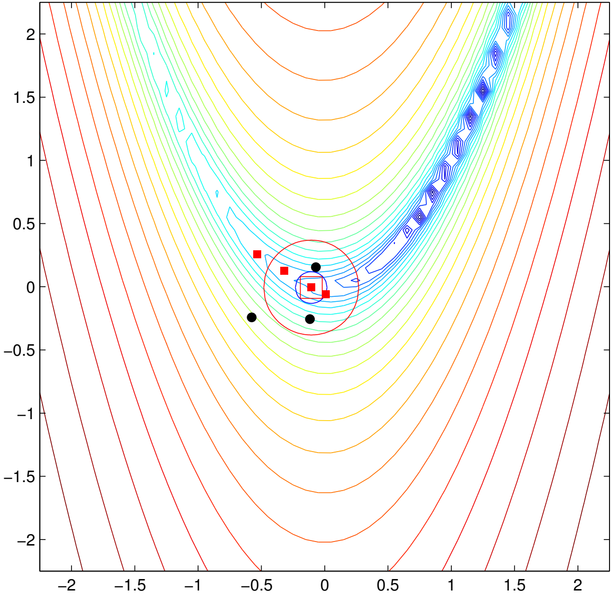
Prototype



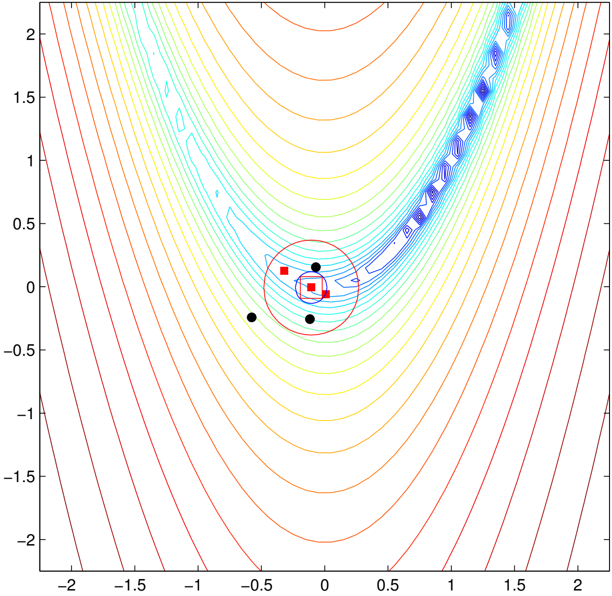
Prototype



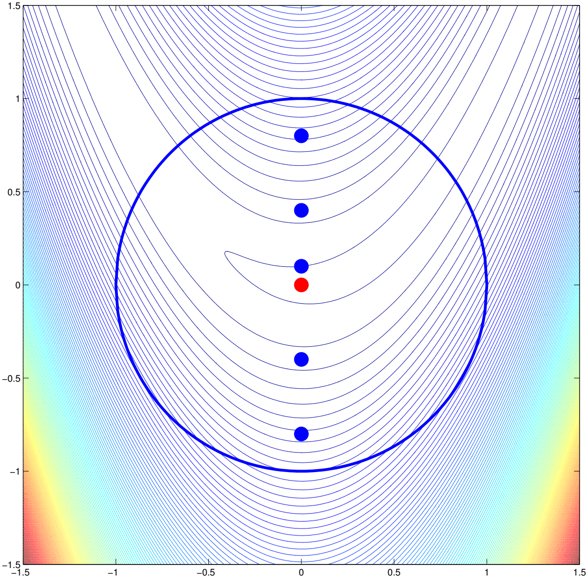
Prototype



Prototype

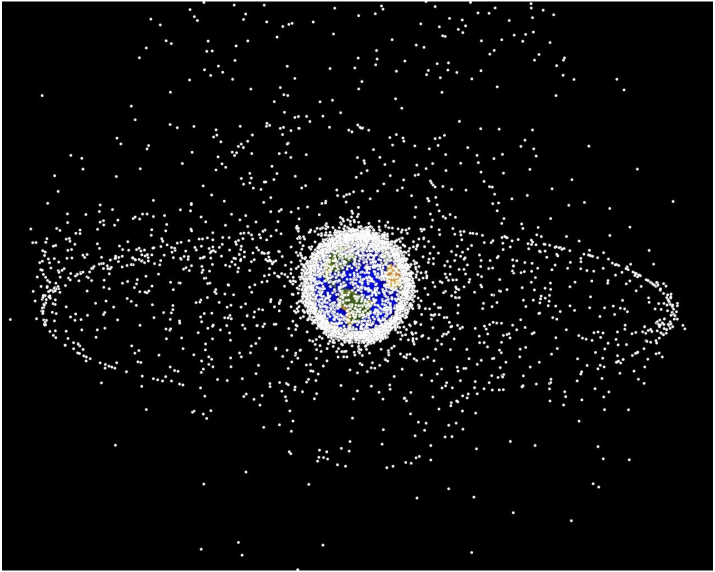


Strongly Λ-poised Sets



Example (Spall)

Example



Courtesy NASA/JPL-Caltech

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$
  - For Kiefer-Wolfowitz,

$$G_i(x^k) = \frac{\bar{f}(x^k + c_k e_i) - \bar{f}(x^k - c_k e_i)}{2c_k}$$

where e_i is the ith column of I_n.

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$
  - For Spall's SPSA,

$$G_i(x^k) = \frac{\bar{f}(x^k + c_k\delta^k) - \bar{f}(x^k - c_k\delta^k)}{2c_k\delta_i^k}$$

where $\delta^k \in \mathbb{R}^n$ is a random perturbation vector

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$
- a_k is a sequence of step sizes

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$

- a_k is a sequence of step sizes (specified by the user) satisfying:

$$\sum_{k=1}^{\infty} a_k = \infty \qquad \lim_{k \to \infty} a_k = 0$$

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$

- a_k is a sequence of step sizes (specified by the user) satisfying:

$$\sum_{k=1}^{\infty} a_k = \infty \qquad \lim_{k\to\infty} a_k = 0$$

Algorithm performance depends significantly on sequence a_k.

Response Surface Methodology

- Developed by the experimental design community.
- Build models using a fixed pattern of points, for example, cubic, spherical, or orthogonal designs among many others.
- Finding the design that constructs response surfaces approximating the function without requiring excessive function evaluations can be difficult for problems where the user has no prior expertise.

Modifications to Existing Methods

Take a favorite method and repeatedly evaluate the function at points of interest.

- Stochastic approximation modified by Dupuis, Simha (1991)
- Response surface methods modified by Chang et al. (2012)
- UOBYQA modified by Deng, Ferris (2006)
- Nelder-Mead modified by Tomick et al. (1995)
- DIRECT modified by Deng, Ferris (2007)

Modifications to Existing Methods

Take a favorite method and repeatedly evaluate the function at points of interest.

- Stochastic approximation modified by Dupuis, Simha (1991)
- Response surface methods modified by Chang et al. (2012)
- UOBYQA modified by Deng, Ferris (2006)
- Nelder-Mead modified by Tomick et al. (1995)
- DIRECT modified by Deng, Ferris (2007)

There are two downsides to such an approach:

1. Repeated sampling provides information about the noise ϵ, not f.
2. If the noise is deterministic, no information is gained.

Overview

We therefore desire a method that

1. Adjusts the step size as it progresses
2. Does not use a fixed design of points
3. Does not repeatedly sample points

Overview

We therefore desire a method that

1. Adjusts the step size as it progresses
2. Does not use a fixed design of points
3. Does not repeatedly sample points

We'd like the class of possible models to be general.

κ-fully Linear model

Definition

If $f \in LC$ and $\exists$ a vector $\kappa = (\kappa_{ef}, \kappa_{eg})$ of positive constants such that

- the error between the gradient of the model and the gradient of the function satisfies

$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_{eg}\Delta \quad \forall y \in B(x;\Delta),$$

- the error between the model and the function satisfies

$$|f(y) - m(y)| \leq \kappa_{ef}\Delta^2 \quad \forall y \in B(x;\Delta),$$

we say the model is κ-*fully linear* on $B(x;\Delta)$.

α-probabilistically κ-fully Linear model

Definition

Let $\kappa = (\kappa_{ef}, \kappa_{eg})$ be a given vector of constants, and let $\alpha \in (0, 1)$. Let $B \subset \mathbb{R}^n$ be given. A random model m_k generated at the kth iteration of an algorithm is α-probabilistically κ-fully linear on B if

$$P\left(m_k \text{ is a } \kappa\text{-fully linear model of } f \text{ on } B \middle| \mathcal{F}_{k-1}\right) \geq \alpha,$$

where $\mathcal{F}_{k-1}$ denotes the realizations of all the random events for the first $k-1$ iterations.

Regression Models can be α-prob. κ-fully Linear

Theorem

For a given $x \in \mathbb{R}^n$, $\Delta > 0$, $\alpha \in (0, 1)$,

- $Y \subset B(x; \Delta)$ *is strongly* Λ*-poised,*
- *The noise present in* $\bar{f}$ *is i.i.d. with mean 0, variance* $\sigma^2 < \infty$,
- $|Y| \geq C/\Delta^4$,

Then there exist constants $\kappa = (\kappa_{ef}, \kappa_{eg})$ *(independent of* Δ *and* Y*) such that the linear model* m *regressing* Y *is* α*-probabilistically* κ*-fully linear on* $B(x; \Delta)$.

Measuring Progress

In traditional trust region methods, if $x^k + s^k$ is the minimizer of m_k, the success of moving from x^k to $x^k + s^k$ is measured by

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

Measuring Progress

In traditional trust region methods, if $x^k + s^k$ is the minimizer of m_k, the success of moving from x^k to $x^k + s^k$ is measured by

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

In the stochastic case, a similar calculation is not obvious.

$$\rho_k = \frac{\bar{f}(x^k) - \bar{f}(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

Measuring Progress

In traditional trust region methods, if $x^k + s^k$ is the minimizer of m_k, the success of moving from x^k to $x^k + s^k$ is measured by

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

In the stochastic case, a similar calculation is not obvious.

$$\rho_k = \frac{m_k(x^k) - m_k(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

Measuring Progress

In traditional trust region methods, if $x^k + s^k$ is the minimizer of m_k, the success of moving from x^k to $x^k + s^k$ is measured by

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

In the stochastic case, a similar calculation is not obvious.

$$\rho_k = \frac{m_k(x^k) - \hat{m}_k(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

Measuring Progress

In traditional trust region methods, if $x^k + s^k$ is the minimizer of m_k, the success of moving from x^k to $x^k + s^k$ is measured by

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

In the stochastic case, a similar calculation is not obvious.

$$\rho_k = \frac{F_k^0 - F_k^s}{m_k(x^k) - m_k(x^k + s^k)}$$

One Last Part

For our analysis, we need estimates of $f(x^k)$ and $f(x^k + s^k)$ that are slightly different than those provided by the model functions.

Let F_k^0 and F_k^s denote the sequence of estimates of $f(x^k)$ and $f(x^k + s^k)$.

We need to be able to construct estimates satisfying

$$\mathbb{P}\left[\left|F_k^0 - f(x^k)\right| > \epsilon \min\left\{\Delta_k, \Delta_k^2\right\} \middle| \mathcal{F}_{k-1}\right] < \theta$$
$$\text{and } \mathbb{P}\left[\left|F_k^s - f(x^k + s^k)\right| > \epsilon \min\left\{\Delta_k, \Delta_k^2\right\} \middle| \mathcal{F}_{k-1}\right] < \theta,$$

for any $\epsilon > 0$ and $\theta > 0$.

Algorithm 1: A trust region algorithm to minimize a stochastic function

Set $k = 0$;

Start

Build a α-probabilistically κ-fully linear model m_k on $B(x^k; \Delta_k)$;

Compute $s^k = \arg \min_{s:\|x^k - s\| \leq \Delta_k} m_k(s)$;

if $m_k(s^k) - m_k(x^k + s^k) \geq \beta \Delta_k$ **then**

  Calculate $\rho_k = \dfrac{F_k^0 - F_k^s}{m_k(x^k) - m_k(x^k + s^k)}$;

  if $\rho_k \geq \eta$ **then**

    Calculate $x^{k+1} = x^k + s^k$; $\Delta_{k+1} = \gamma_{inc} \Delta_k$;

  else

    $x^{k+1} = x^k$; $\Delta_{k+1} = \gamma_{dec} \Delta_k$;

  end

else

  $x^{k+1} = x^k$; $\Delta_{k+1} = \gamma_{dec} \Delta_k$;

end

$k = k + 1$ and go to **Start**;

Convergence

Under what assumptions will our algorithm converge almost surely to a first-order stationary point?

- Assumptions on f
- Assumptions on ϵ
- Assumptions on algorithmic constants

Convergence

Assumption

On some set $\Omega \subseteq \mathbb{R}^n$ containing all iterates visited by the algorithm,

- *f is Lipschitz continuous*
- *∇f is Lipschitz continuous*
- *f has bounded level sets*

Assumption

The additive noise ϵ observed when computing $\bar{f}$ is independent and identically distributed with mean zero and bounded variance σ^2.

Convergence

Assumption

The constants $\alpha \in (0,1)$*,* $\gamma_{dec} \in (0,1)$*, and* $\gamma_{inc} > 1$ *satisfy*

$$\alpha \geq \max\left\{\frac{1}{2}, 1 - \frac{\frac{\gamma_{inc}-1}{\gamma_{inc}}}{4\left[\frac{\gamma_{inc}-1}{2\gamma_{inc}} + \frac{1-\gamma_{dec}}{\gamma_{dec}}\right]}\right\},$$

where

- α *is the lower bound on the probability of having a* κ*-fully linear model,*
- $\gamma_{dec} \in (0,1)$ *is the factor by which we decrease the trust region radius,*
- $\gamma_{inc} > 1$ *is the factor by which the trust radius is increased.*

Convergence

Assumption

The constants $\alpha \in (0,1)$, $\gamma_{dec} \in (0,1)$, and $\gamma_{inc} > 1$ satisfy

$$\alpha \geq \max \left\{ \frac{1}{2}, 1 - \frac{\frac{\gamma_{inc}-1}{\gamma_{inc}}}{4\left[\frac{\gamma_{inc}-1}{2\gamma_{inc}} + \frac{1-\gamma_{dec}}{\gamma_{dec}}\right]} \right\},$$

where

- α *is the lower bound on the probability of having a* κ*-fully linear model,*
- $\gamma_{dec} \in (0,1)$ *is the factor by which we decrease the trust region radius,*
- $\gamma_{inc} > 1$ *is the factor by which the trust radius is increased.*

If $\gamma_{inc} = 2$ and $\gamma_{dec} = 0.5 \rightarrow \alpha \geq 0.9$
If $\gamma_{inc} = 2$ and $\gamma_{dec} = 0.9 \rightarrow \alpha \geq 0.65$

Proof Outline

Theorem

If the above assumptions are satisfied, our algorithm converges almost surely to a first-order stationary point of f.

- Show the sequence of trust region radii $\Delta_k \to 0$ almost surely.

Proof Outline

Theorem

If the above assumptions are satisfied, our algorithm converges almost surely to a first-order stationary point of f.

- Show the sequence of trust region radii $\Delta_k \to 0$ almost surely.
- Show if Δ_k ever falls below some constant multiple of the model gradient, $\Delta_{k+1} > \Delta_k$ with high probability.

Proof Outline

Theorem

If the above assumptions are satisfied, our algorithm converges almost surely to a first-order stationary point of f.

- Show the sequence of trust region radii $\Delta_k \to 0$ almost surely.
- Show if Δ_k ever falls below some constant multiple of the model gradient, $\Delta_{k+1} > \Delta_k$ with high probability.
- Lastly, show that, the sequence of ratios

$$\left\{ \frac{\left\|\nabla f(x^k)\right\|}{\Delta_k} \right\}$$

is bounded above by a nonnegative supermartingale. Since every nonnegative supermartingale converges almost surely, and $\Delta_k \to 0$ almost surely, this implies $\left\|\nabla f(x^k)\right\| \to 0$ almost surely.

Algorithm 1: A trust region algorithm to minimize a stochastic function

Set $k = 0$;

Start

Build a α-probabilistically κ-fully linear model m_k on $B(x^k; \Delta_k)$;

Compute $s^k = \arg \min_{s:\|x^k - s\| \leq \Delta_k} m_k(s)$;

if $m_k(s^k) - m_k(x^k + s^k) \geq \beta \Delta_k$ **then**

  Calculate $\rho_k = \dfrac{F_k^0 - F_k^s}{m_k(x^k) - m_k(x^k + s^k)}$;

  if $\rho_k \geq \eta$ **then**

    Calculate $x^{k+1} = x^k + s^k$; $\Delta_{k+1} = \gamma_{inc} \Delta_k$;

  else

    $x^{k+1} = x^k$; $\Delta_{k+1} = \gamma_{dec} \Delta_k$;

  end

else

  $x^{k+1} = x^k$; $\Delta_{k+1} = \gamma_{dec} \Delta_k$;

end

$k = k + 1$ and go to **Start**;

Prototype

- m_k is a linear regression model on a sample set of $(n+1)C_k$ sample points, where C_k is defined by

$$C_k = \left\lceil \frac{k}{1000} \right\rceil \frac{\max\left\{n+1, \left\lfloor \frac{1}{\Delta_k^4} \right\rfloor\right\}}{n+1}.$$

The sample set consists of C_k randomly rotated copies of the set

$$\{x^k, x^k + \Delta_k e_1, \ldots, x^k + \Delta_k e_n\}$$

Prototype

- m_k is a linear regression model on a sample set of $(n+1)C_k$ sample points, where C_k is defined by

$$C_k = \left\lceil \frac{k}{1000} \right\rceil \frac{\max\left\{n+1, \left\lfloor \frac{1}{\Delta_k^4} \right\rfloor\right\}}{n+1}.$$

  The sample set consists of C_k randomly rotated copies of the set

$$\{x^k, x^k + \Delta_k e_1, \ldots, x^k + \Delta_k e_n\}$$

- $F_k^0 = m_k^0(x^k)$, where m_k^0 is a linear regression model using C_k randomly rotated copies of the set

$$\{x^k, x^k + 0.5\Delta_k e_1, \ldots, x^k + 0.5\Delta_k e_n\}$$

Prototype

- m_k is a linear regression model on a sample set of $(n+1)C_k$ sample points, where C_k is defined by

$$C_k = \left\lceil \frac{k}{1000} \right\rceil \frac{\max\left\{n+1, \left\lfloor \frac{1}{\Delta_k^4} \right\rfloor\right\}}{n+1}.$$

  The sample set consists of C_k randomly rotated copies of the set

$$\{x^k, x^k + \Delta_k e_1, \ldots, x^k + \Delta_k e_n\}$$

- $F_k^s = m_k^s(x^k)$, where m_k^s is a linear regression model using C_k randomly rotated copies of the set

$$\{x^k + s^k, x^k + s^k + 0.5\Delta_k e_1, \ldots, x^k + s^k + 0.5\Delta_k e_n\}$$

Problem Set

53 problems of the form:

$$f(x) = \sum_{i=1}^{m} \left[(1+\sigma)F_i(x)\right]^2,$$

where $\sigma \sim U[-0.1, 0.1]$.

Problem Set

53 problems of the form:

$$f(x) = \sum_{i=1}^{m} \left[(1+\sigma)F_i(x)\right]^2,$$

where $\sigma \sim U[-0.1, 0.1]$.

If S is the set of solvers to be compared on a suite of problems P, let $t_{p,s}$ be the number of iterates required for solver $s \in S$ on a problem $p \in P$ to find a function value satisfying:

$$f(x) - f_L \leq \tau \left(f(x^0) - f_L\right),$$

where f_L is the best function value achieved by any $s \in S$.

Problem Set

Comments

- We are using the true function value f, not the observed $\bar{f}$.
- Since the noise is stochastic, each solver is run 10 times per problem.

If S is the set of solvers to be compared on a suite of problems P, let $t_{p,s}$ be the number of iterates required for solver $s \in S$ on a problem $p \in P$ to find a function value satisfying:

$$f(x) - f_L \leq \tau \left(f(x^0) - f_L \right),$$

where f_L is the best function value achieved by any $s \in S$.

Performance Profile

Then the performance profile of a solver $s \in S$ is the following fraction:

$$\rho_s(\phi) = \frac{1}{|P|} \left| \left\{ p \in P : \frac{t_{p,s}}{\min \{ t_{p,s} : s \in S \}} \leq \phi \right\} \right|$$

Performance Profile

Then the performance profile of a solver $s \in S$ is the following fraction:

$$\rho_s(\phi) = \frac{1}{|P|} \left| \left\{ p \in P : \frac{t_{p,s}}{\min \{ t_{p,s} : s \in S \}} \leq \phi \right\} \right|$$

- $\rho_s(1)$: Fraction of P method s solves first.
- $\lim_{\phi \to \infty} \rho_s(\phi)$: Fraction of P method s eventually solves.
- $\rho_s(\phi)$: Fraction of P method s solves in under ϕ times the evaluations required for the best method.

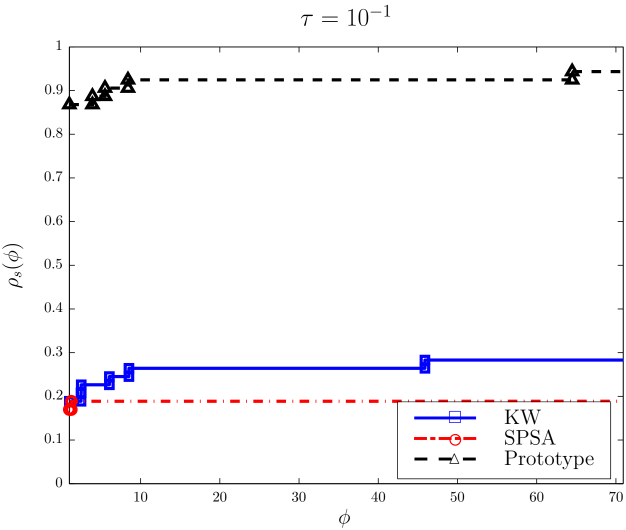
Performance Profile

We compare our prototype against Spall's versions of Kiefer-Wolfowitz and SPSA with step sizes as recommended in Sections 6.6 and 7.5.2 of Spall (2003)

$$a_k = \frac{1}{(k+1+A)^{0.602}} \qquad c_k = \frac{1}{(k+1)^{0.101}}$$

where A is one tenth of the total budget of function evaluations.

Performance Profile



$\rho_s(1)$:

Fraction s solves first

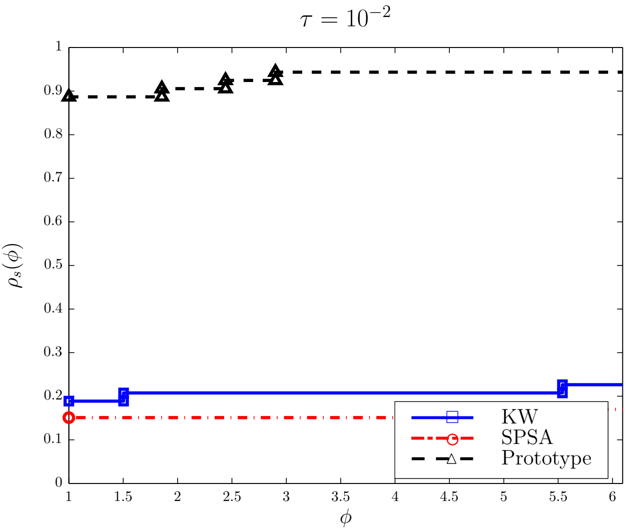
$\lim_{\phi\to\infty} \rho_s(\phi)$:

Fraction s solves

$\rho_s(\phi)$:

Fraction s solves in under ϕ times the evaluations required for the best method.

Performance Profile



$\rho_s(1)$:

Fraction s solves first

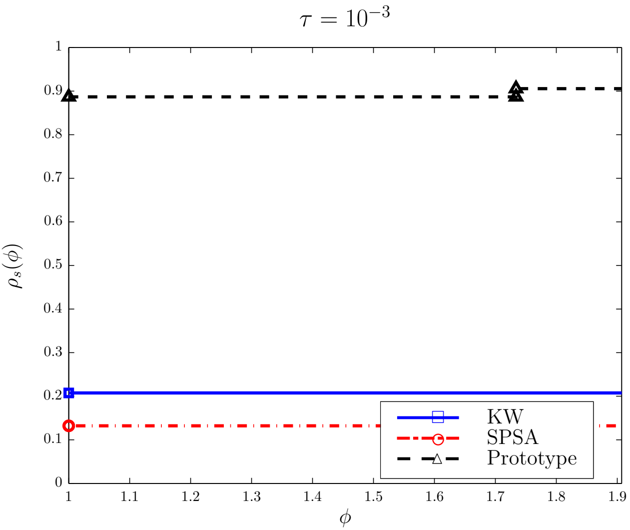
$\lim_{\phi\to\infty} \rho_s(\phi)$:

Fraction s solves

$\rho_s(\phi)$:

Fraction s solves in under ϕ times the evaluations required for the best method.

Performance Profile



$\rho_s(1)$:

Fraction s solves first

$\lim_{\phi\to\infty} \rho_s(\phi)$:

Fraction s solves

$\rho_s(\phi)$:

Fraction s solves in under ϕ times the evaluations required for the best method.

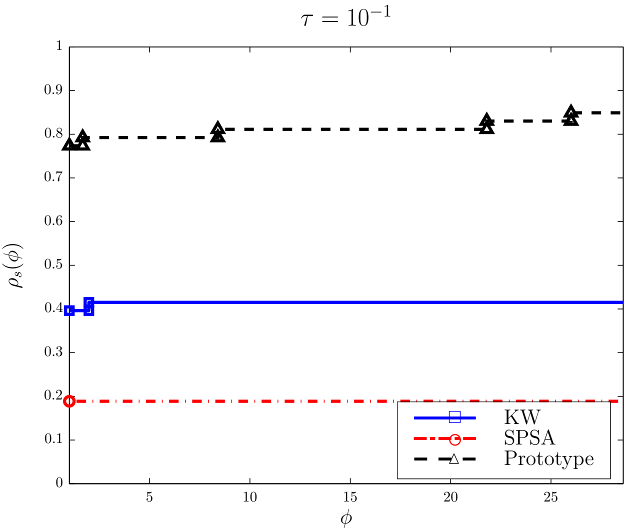
Another Problem Set

53 problems of the form:

$$f(x) = \sigma_p + \sum_{i=1}^{m} [F_i(x)]^2 ,$$

where $\sigma_p \sim N\left(0, (0.1\Delta_p)^2\right)$ **and** $\Delta_p = \sum_i F_i(x^0) - \sum_i F_i(x^*)$**.**

Performance Profile



$\rho_s(1)$:

Fraction s solves first

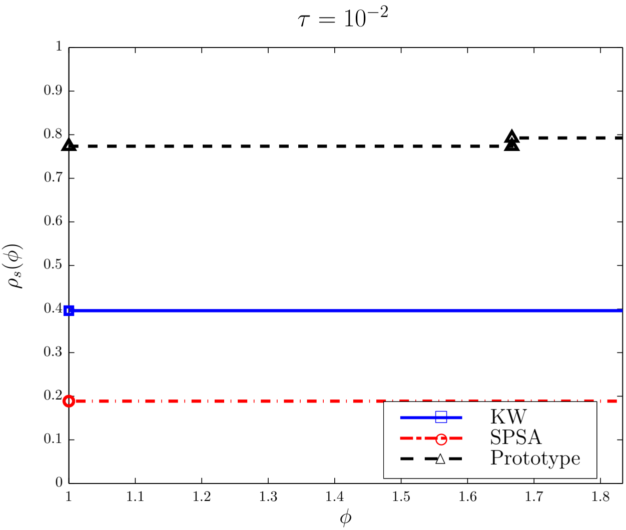
$\lim_{\phi\to\infty} \rho_s(\phi)$:

Fraction s solves

$\rho_s(\phi)$:

Fraction s solves in under ϕ times the evaluations required for the best method.

Performance Profile



$\rho_s(1)$:

Fraction s solves first

$\lim_{\phi \to \infty} \rho_s(\phi)$:

Fraction s solves

$\rho_s(\phi)$:

Fraction s solves in under ϕ times the evaluations required for the best method.

Further Information and Current Work

Preprint on Optimization Online

"Stochastic Derivative-free Optimization using a Trust Region Framework"

Code

`http://people.kth.se/~jeffreyl/Stochastic/`

Further Information and Current Work

Preprint on Optimization Online

"Stochastic Derivative-free Optimization using a Trust Region Framework"

Code

`http://people.kth.se/~jeffreyl/Stochastic/`

- Generalizing results to ensure a practical algorithm converges.
  - For example, not requiring α-probabilistically κ-fully linear models every iteration.

Further Information and Current Work

Preprint on Optimization Online

"Stochastic Derivative-free Optimization using a Trust Region Framework"

Code

http://people.kth.se/∼jeffreyl/Stochastic/

- Generalizing results to ensure a practical algorithm converges.
  - For example, not requiring α-probabilistically κ-fully linear models every iteration.
- Smartly constructing α-probabilistically κ-fully linear models.

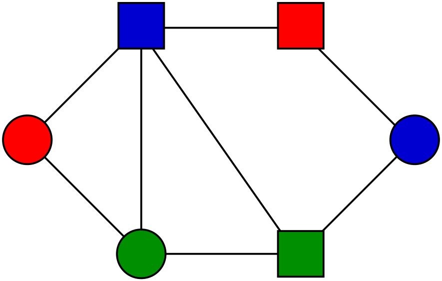
Outline

Contents

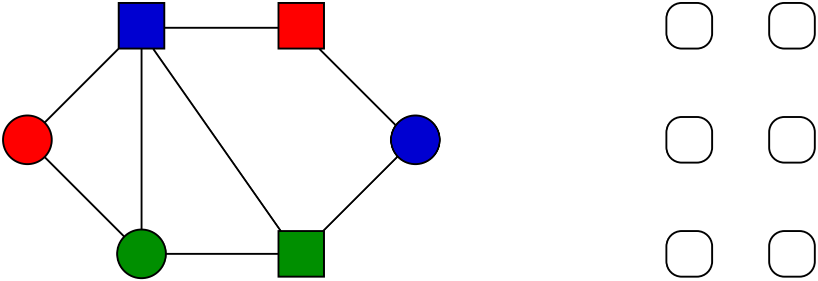
Joint work with Euhanna Ghadimi and Mikael Johansson

Motivation

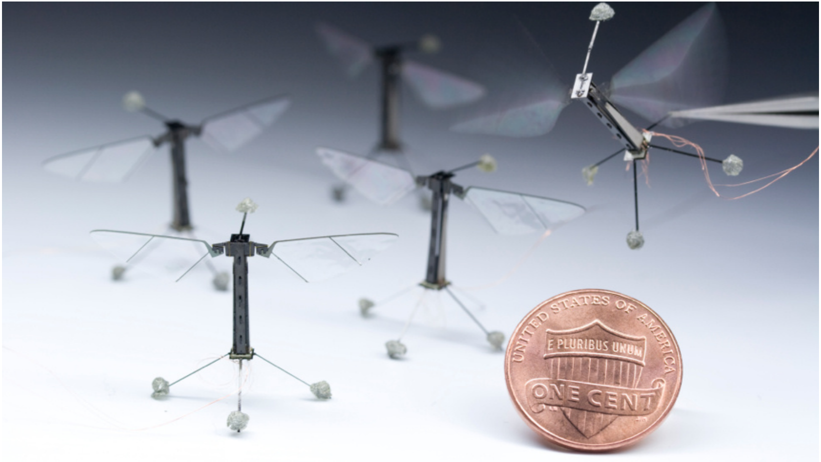
Motivation



Motivation



Motivation



Credit: ***RoboBees Project, Harvard University***

Problem Statement

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \mathbb{E}\left[\sum_{i=1}^{N} \bar{f}_i(x)\right] \\ \text{subject to} \quad & Ax \leq b \\ & x \in X \end{aligned}$$

- Each agent has objective $f_i(x)$ which can only be observed with additive noise $\bar{f}_i(x) = f_i(x) + \epsilon$
- Each f_i is convex
- ϵ has zero mean and finite variance
- X is an is a nonempty, closed, convex subset of $\mathbb{R}^n$

Algorithm

Goal

Agents connected by a network cooperatively minimize the global objective though they only have knowledge of their individual objectives (and shared information from the network).

- Aim: Distributed Multi-agent Derivative-free Optimization
  - At iteration j, agent i builds a model m_j^i using observed values of $\bar{f}_i$.
  - Communicate where they are going to their neighbors in the network.
  - Take the information from their neighbors for iteration $j+1$.

Algorithm

Goal

Agents connected by a network cooperatively minimize the global objective though they only have knowledge of their individual objectives (and shared information from the network).

- Aim: Distributed Multi-agent Derivative-free Optimization
  - At iteration j, agent i builds a model m_j^i using observed values of $\bar{f}_i$.
  - Communicate where they are going to their neighbors in the network.
  - Take the information from their neighbors for iteration $j+1$.
- Today: Distributed Multi-agent Optimization with Inexact Subproblems

Problem Statement

$$\underset{x}{\text{minimize}} \quad \sum_i f_i(x)$$

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_i f_i(x_i) \\ \text{subject to} \quad & x_i = x_j \qquad \forall (i,j) \in \mathcal{E} \end{aligned}$$

Problem Statement

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & Ax \le b \\ & x \in X \end{aligned}$$

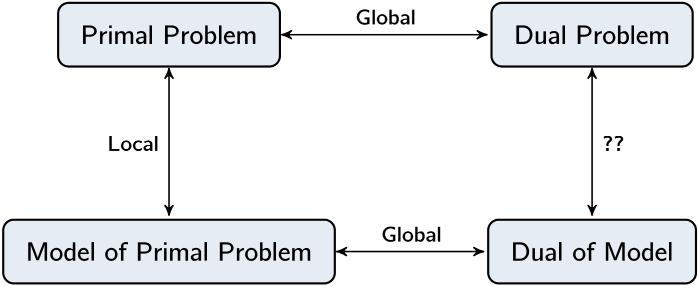
or

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned}$$

Previous Methods

Lagrangian dual decomposition methods (Nedić, Ozdaglar, Johansson...)

- Challenge for using the dual when constructing models:



Previous Methods

Primal Methods using Consensus (Tsitsiklis, Bertsekas)

- Can be quite slow

Iterates have the form:

$$x_i^{k+1} = \sum_{j=1}^{N} w_{ij} x_j^k - a d_i^k$$

where a is a step size, d_i^k is an element of the subdifferential of f_i at x_i^k.

Previous Methods

Primal Methods using Consensus (Tsitsiklis, Bertsekas)

- Can be quite slow

Iterates have the form:

$$x_i^{k+1} = \sum_{j=1}^{N} w_{ij} x_j^k - a d_i^k$$

where a is a step size, d_i^k is an element of the subdifferential of f_i at x_i^k.

$$f(y^k) \leq f^* + aL^2 C_1 + \frac{NLBC_2}{k} \sum_{j=1}^{N} \left\| x_j^0 \right\| + \frac{N}{2ak} \left(\mathrm{dist}(y^0, X^*) + aL \right)^2$$

Nedić, Ozdaglar (2009)

Previous Methods

Alternating Direction Method of Multipliers (ADMM)

- Developed in the 1970s (Hestenes, Powell, Eckstein)
- Roots in the 1950s (Dantzig, Wolfe, Benders)
- Equivalent or similar to many other algorithms

Previous Methods

Alternating Direction Method of Multipliers (ADMM)

- Developed in the 1970s (Hestenes, Powell, Eckstein)
- Roots in the 1950s (Dantzig, Wolfe, Benders)
- Equivalent or similar to many other algorithms
  - Douglas-Rachford splitting
  - Spingarn's method of partial inverses
  - Dykstra's alternating projections
  - Proximal methods
  - Bregman iterative methods
  - More...

ADMM

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned} \tag{1}$$

has augmented Lagrangian

$$L_\rho(x, z, \mu) = f(x) + g(z) + \mu^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2$$

ADMM

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned} \tag{1}$$

has augmented Lagrangian

$$L_\rho(x, z, \mu) = f(x) + g(z) + \mu^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2$$

Algorithm 2: Traditional ADMM

Pick initial values z^0, μ^0, ρ;
for $k = 0, 1, \ldots$ **do**
$\quad x^{k+1} = \arg\min_x L_\rho(x, z^k, \mu^k)$;
$\quad z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \mu^k)$;
$\quad \mu^{k+1} = \mu^k + \rho \left(Ax^{k+1} + Bz^{k+1} - c\right)$;
end

Previous inexact ADMM methods

Algorithm 3: Deng, Yin (2013) Generalized ADMM

Pick $Q \succeq 0$ and symmetric P, z^0, μ^0, ρ;

for $k = 0, 1, \ldots$ **do**

$x^{k+1} = \arg\min_x L_\rho(x, z^k, \mu^k) + \frac{1}{2}(x - x^k)P(x - x^k)$;

$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \mu^k) + \frac{1}{2}(z - z^k)Q(z - z^k)$;

$\mu^{k+1} = \mu^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right)$;

end

Previous inexact ADMM methods

Algorithm 3: Deng, Yin (2013) Generalized ADMM

Pick $Q \succeq 0$ and symmetric P, z^0, μ^0, ρ;

for $k = 0, 1, \dots$ **do**

$x^{k+1} = \arg\min_x L_\rho(x, z^k, \mu^k) + \frac{1}{2}(x - x^k)P(x - x^k)$;

$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \mu^k) + \frac{1}{2}(z - z^k)Q(z - z^k)$;

$\mu^{k+1} = \mu^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right)$;

end

- Fixed matrices P and Q
- Still dealing with $\arg\min_x f$

Our approach

Algorithm 4: Our modification of ADMM

Pick initial values z^0, μ^0, ρ;

for $k = 0, 1, 2, \dots$ **do**

$$x^{k+1} = \arg\min_x f(x^k) + \nabla_x f(x^k)^T (x - x^k) + \frac{1}{2}(x - x^k)^T \nabla_x^2 f(x^k)(x - x^k) + (\mu^k)^T (Ax + Bz^k - c) + \frac{\rho}{2} \left\| Ax + Bz^k - c \right\|;$$

$$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \mu^k);$$

$$\mu^{k+1} = \mu^k + \rho \left(Ax^{k+1} + Bz^{k+1} - c \right);$$

end

Our approach

Algorithm 4: Our modification of ADMM

Pick initial values z^0, μ^0, ρ;

for $k = 0, 1, 2, \ldots$ **do**

$$x^{k+1} = \arg\min_x f(x^k) + \nabla_x f(x^k)^T (x - x^k) + \frac{1}{2}(x - x^k)^T \nabla_x^2 f(x^k)(x - x^k) + (\mu^k)^T (Ax + Bz^k - c) + \frac{\rho}{2} \left\| Ax + Bz^k - c \right\|;$$

$$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \mu^k);$$

$$\mu^{k+1} = \mu^k + \rho \left(Ax^{k+1} + Bz^{k+1} - c \right);$$

end

Assumption

Assume f is convex and twice continuously differentiable in the region of interest so $\nabla^2 f(x^k)$ is well-defined.

Convergence

Assumption

There exists a saddle point to problem (1). In other words, there exists points x^*, z^*, μ^* satisfying

$$\begin{aligned} \nabla_z g(z^*) + B^T \mu^* &= 0 \\ \nabla_x f(x^*) + A^T \mu^* &= 0 \\ Ax^* + Bz^* &= c \end{aligned}$$

Define $\|x\|_A^2 = x^T A x$ and

$$y^* = \begin{bmatrix} x^* \\ z^* \\ \mu^* \end{bmatrix}, y^k = \begin{bmatrix} x^k \\ z^k \\ \mu^k \end{bmatrix}, H_k = \begin{bmatrix} \nabla_x^2 f(x^k) + \rho A^T A & 0 & 0 \\ 0 & \rho I & 0 \\ 0 & 0 & \frac{1}{\rho} I \end{bmatrix}$$

Convergence

Lemma

Iterates generated by our algorithm satisfy

$$\left\|y^k - y^*\right\|_{H_k}^2 - \left\|y^{k+1} - y^*\right\|_{H_k}^2 \geq \left\|x^k - x^{k+1}\right\|_{(\nabla_x^2 f(x^k) + \rho A^T A - \frac{1}{\beta} A^T A)}^2 + \left(\frac{1}{\rho} - \beta\right) \left\|\mu^k - \mu^{k+1}\right\|^2$$

for all $\beta > 0$*.*

Convergence

Lemma

Iterates generated by our algorithm satisfy

$$\left\|y^k - y^*\right\|_{H_k}^2 - \left\|y^{k+1} - y^*\right\|_{H_k}^2 \geq \left\|x^k - x^{k+1}\right\|_{(\nabla_x^2 f(x^k) + \rho A^T A - \frac{1}{\beta} A^T A)}^2 + \left(\frac{1}{\rho} - \beta\right) \left\|\mu^k - \mu^{k+1}\right\|^2$$

for all $\beta > 0$.

- This shows y^k converges to y^* if $\nabla^2 f(x^k) \succ 0$.

Convergence

Lemma

Iterates generated by our algorithm satisfy

$$\left\|y^k - y^*\right\|_{H_k}^2 - \left\|y^{k+1} - y^*\right\|_{H_k}^2 \geq \left\|x^k - x^{k+1}\right\|^2_{(\nabla_x^2 f(x^k) + \rho A^T A - \frac{1}{\beta} A^T A)} + (\frac{1}{\rho} - \beta)\left\|\mu^k - \mu^{k+1}\right\|^2$$

for all $\beta > 0$.

- This shows y^k converges to y^* if $\nabla^2 f(x^k) \succ 0$.
- This shows y^k converges to some $\bar{y}$ if $\nabla^2 f(x^k) \succeq 0$.

Example: General ℓ_1 Regularized Loss Minimization

Consider the problem

$$\text{minimize } l(x) + \lambda \|x\|_1$$

where l is any convex loss function. In ADDM form, we can write this:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & l(x) + g(z) \\ \text{subject to} & x - z = 0 \end{array}$$

where $g(z) = \|z\|_1$.

Example: General ℓ_1 Regularized Loss Minimization

Consider the problem

$$\text{minimize}\, l(x) + \lambda \|x\|_1$$

where l is any convex loss function. In ADDM form, we can write this:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & l(x) + g(z) \\ \text{subject to} & x - z = 0 \end{array}$$

where $g(z) = \|z\|_1$.

- Instead of solving the x-update exactly, solving the quadratic approximation can be faster.

Results

$$\text{minimize} \sum \left(\log\left(-b_i(a_i^T x)\right)\right) + \lambda \|x\|_1$$

where a_i are rows in a feature matrix A and b is a response vector.

- Boyd's exact minimization (for a large problem) takes a total of 4928 iterations (summing over all agents)
- Solving only a single Newton step takes 1700 iterations

Concerns and Assumptions

Concerns

- Time varying network
- Asynchronous updates
- Delays in communication
- Nonconvex local objectives

Concerns and Assumptions

Concerns

- Time varying network
- Asynchronous updates
- Delays in communication
- Nonconvex local objectives

Assumptions

- Constant network
- Synchronized updates
- No delays in communication
- Convex local objectives

Thanks

Questions?

jeffreyl@kth.se